# Three concepts for light-weight communication in multiplayer games

Jouni Smed and Harri Hakonen

Department of Information Technology, FI-20014 University of Turku, Finland
jouni.smed@utu.fi, harri.hakonen@utu.fi

**Abstract.** We introduce three game design concepts which, at the same time, allow interaction among multiple players but do not require as large networking resources as real-time interaction. These concepts can be used, for example, in mobile platforms where the communication resources are limited.

## 1 Introduction

Achieving real-time response is still a major hurdle for mobile platforms, because of their limited processing power, memory capacity, display capabilities, and communication channels. Especially, the resources for handling the network communication – bandwidth, latency, and the node's processing power – impose restrictions on real-time communication, which the application cannot overcome and which must be considered in the game design [1, Chap. 8]. Although we can wait and hope for improvements on the underlying technology, this article takes a more proactive view and introduces game design concepts which use light-weight communication.

Real-time communication is not the only method to allow multiple players to participate in a game simultaneously. For example, the oldest form of non-real-time multiplaying, dating back from the 1970s, is a high-score list which provides an after-game place for the players to meet and compete with their results. It is still a viable form of interaction, and the competition can be distributed so that the participants provide their results which are then compiled to form the final standings [2].

The three concepts presented in this paper are examples of decision-making in operational, tactical and strategic level [1, Chap. 6]. They correspond to the time span and abstractness of the decisions: operational decisions are concrete and frequently issued commands, tactical decisions comprise instructions aimed at a given situation, and strategic decisions focus on long-term planning. We line out the requirements that each concept imposes to the light-weight communication so that the game remains enjoyable for everyone to play.

## 2 Operational level: Short active turns

The simplest way to achieve interaction is to serialize the game events so that each player has a turn when to make decisions. Thus, in a turn-based game the

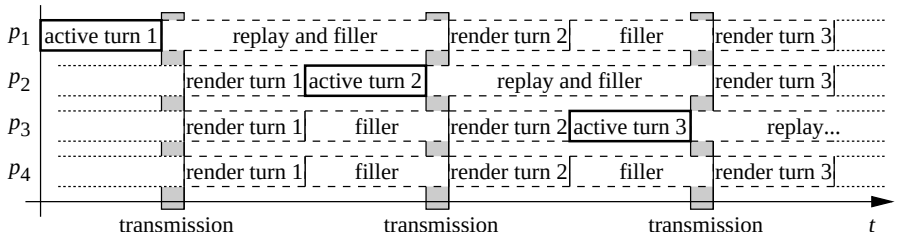


**Fig. 1.** Each player has successively a short active turn followed by passive turns, where the other players make their attempts.

players have active turns followed by passive turns where they are observing the game progress. If the player's decisions are carried out immediately, this active turn cannot be too long so that all the players have an equal chance to interact and the waiting times remain reasonable. Whereas the active turns matter the most to the player, we should smooth out their difference to the passive ones. This means that the game design should make the players also find the compulsory passive turns interesting and captivating.

Figure 1 illustrates how the game-play works: Each turn has a predefined length. When the active player is making an attempt, the passive players can view statistics, prepare for their turn, or customize the presentation of the game content (e.g. the type of the shown filler material). When the active player has completed her attempt, she can watch a replay or post-attempt animation or get comments from the coach. In the meanwhile, the relevant data is transferred to the other players, who can then render it. There is no need for a server but the communication use peer-to-peer architecture. However, joining requires a way to handle the participation management so that the players can connect one another.

The main requirement for such a game design is that the player's operational decisions are made within short time intervals (i.e. active turns). Natural candidates for this kind of a game are certain fast-paced and attempt-based sports events such as javelin, long jump, ski jump, or darts. Such games also retain the excitement in the passive turns, because it is interesting to watch and anticipate other players' attempts. Moreover, this makes it easy to generate relevant filler material (e.g. statistics or slow motion replays) to be shown during the passive turns.

## 3 Tactical level: Semi-autonomous avatars

Instead of operational commands, we can raise the abstraction level to tactical commands, which means that they are not so time-sensitive. This demonstrates the idea of compensating communication with computation. Let us elaborate by using a simple shooter as an example. Rather than giving commands like 'move forward', 'turn left' or 'shoot', which require prompt communication, the

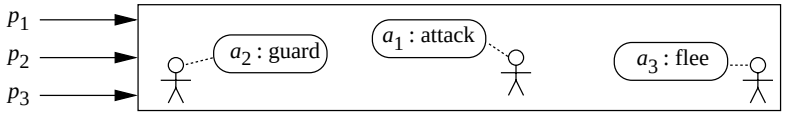


**Fig. 2.** Players $p_1$, $p_2$ and $p_3$ issue tactical instructions to the corresponding avatars $a_1$, $a_2$ and $a_3$.

avatars can be semi-autonomous, to whom the players give tactical instructions like 'attack', 'flee' or 'guard' (see Fig. 2). The avatars then carry out these tactics the best they can. However, their response is not immediate and the outcome can be something else than the player expected. This resembles the characters of *The Sims* [3], which have a limited free will to carry out the player's commands.

Semi-autonomous avatars provide a way to realize light-weight communication in client–server architecture. The players (i.e. clients) send tactical commands to the server, which updates the situation and returns the game events. High latency can be compensated by slowing down the pace of the game or by gathering the commands of a certain period and issuing them simultaneously like the SMS television games [4]. Because the computational burden lies now in the server, we can even allow the players to code the operational level logic themselves like in *Core War* [5] or *AIsHockey* [6].

To summarize, this concept requires that the game design has a clear separation between tactical and operational level. In order to have intelligent avatars the game world should be non-complex (e.g. a limited arena) or the set of actions in the tactical level should be limited. For example, team sports games provide natural command interfaces that accept tactical commands like 'attack on the right side', 'defence go forward', or 'increase pressure'.

## 4 Strategic level: Interaction via proxies

The game-play on the strategic level does not require that the participating players are present at the same time. The players can set proxies that later on interact with other players on their behalf, and, conversely, they encounter proxies set by other players. For example, the bone files of *NetHack* [7] allow the player to interact with the ghost of another player, who has died earlier on that level. The ghost acts then as a proxy for the deceased player, but that player himself do not interact the active player. In addition to fully-autonomous avatars, the proxies can be game entities (e.g. mechanistic objects or gizmos) and they can even include programmable parts.

As an example let us introduce a novel game called *Entrappers* (see Fig. 3). The game comprises levels generated and stored in a server. When a player enters a level, she gets either a computer-generated or previously stored level. A stored level can contain traps set by previous players (i.e. the traps acts as their proxies). The player is alone in the level and only when she exits, the level – containing the possible modifications and new traps she has set – is stored

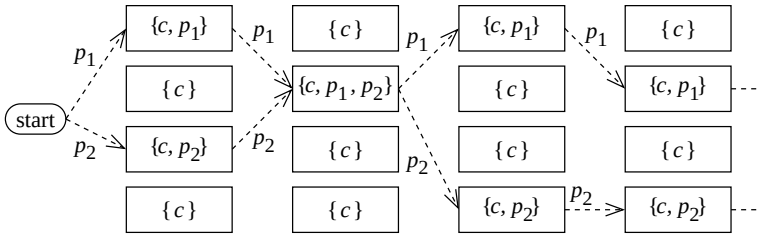


**Fig. 3.** The columns represent level alternatives for a player. The arrows indicate the routes already taken by players $p_1$ and $p_2$. Inside each level are traps set by computer ($c$) or the players.

back to the server. The player is awarded immediately for clearing the level and indirectly over time when somebody else falls into a trap set by her.

This concept requires that the game design allows unrestricted play time for the players (i.e. they can join and leave whenever they want and the rewards are collected over time). Moreover, the game-play lacks immediate human interaction, which also restricts the game design. However, from these restrictions follow that we can implement a light-weight communication that allows to organize games with massively multiple players.

## 5 Concluding remarks

Although we cannot escape technical limitations, we can change the resource requirements by altering the game design cleverly. By considering the different decision-making levels we described concepts that can steer the game design so that we can combine the need for multiplayer support with light-weight communication. Future work is needed to evaluate what kind of games these concepts allow to develop and what kinds of multiplayer gaming needs they can fulfil.

## References

1. Smed, J., Hakonen, H.: Algorithms and Networking for Computer Games. John Wiley & Sons, Chichester, UK (2006)
2. Könönen, V.: Ski Jump International v3. (2006) ⟨http://www.nomasi.com/sj3/⟩.
3. Maxis: The Sims. Electronic Arts (2000) ⟨http://www.maxis.com/⟩.
4. Seppänen, A.: Regional case study: Finland. In: Game Developer Conference Mobile 2003 Proceedings, San Jose, CA (2003) ⟨http://www.gamasutra.com/features/gdcarchive/2003M/Seppanen_Antti.ppt⟩.
5. Dewdney, A.K.: Computer recreations: In the game called Core War hostile programs engage in a battle of bits. Scientific American **250**(5) (1984) 14–22
6. Smed, J., Kaukoranta, T., Hakonen, H.: AIsHockey—a platform for studying synthetic players. In Sing, L.W., Man, W.H., Wai, W., eds.: Proceedings of the 2nd International Conference on Application and Development of Computer Games, Hong Kong SAR, China (2003) 183–188
7. DevTeam: NetHack 3.4.3. (2006) ⟨http://www.nethack.org/⟩.